Bruno Manuel Zamora Garcia A01798275

Actividad 2 (Análisis de transformaciones)

Empezar tarea

Aclaracion: las variables, matrices o vectores que tengan la terminacian "_a" o "'_antropomorfico" y "-p" o "_planar "son varibales que perteneecen al analisis del robot antropomorfico y planar respectivamente.Ademas a lo largo del codigo hay pequenas explicaciones sobre cada parte del codigo

al final del codigo esta el analisis mas detallado.

Aclaracion2: En el pdf no se ven completas mis tabla comparativas, le sugiero que abra el archivo .mlx, por favor.

```
clear all
close all
clc

%% Declaración de variables simbólicas para cada robot
% Robot Planar (sufijo _p)
syms th1_p(t) th2_p(t) th3_p(t) t l1_p l2_p l3_p
% Robot Antropomórfico (sufijo _a)
syms th1_a(t) th2_a(t) th3_a(t) t l1_a l2_a l3_a

%% Configuración de las juntas (0 indica junta rotacional)
RP_planar = [0 0 0];
RP_antropomorfico = [0 0 0];

%% Vectores de coordenadas articulares y sus derivadas
% Estos vectores representan los ángulos de cada junta para cada robot.
Q_planar = [th1_p, th2_p, th3_p];
Q_antropomorfico = [th1_a, th2_a, th3_a];

% Velocidades articulares (derivadas temporales)
Qp_planar = diff(Q_planar, t);
Qp_antropomorfico = diff(Q_antropomorfico, t);

%% Número de grados de libertad (3 GDL para ambos robots)
GDL = size(RP_planar,2);
```

========== ROBOT PLANAR DE 3 GDL ==========

Transformación Local (Planar):

En el robot planar, cada eslabón tiene:

- Rotación pura sobre el eje z, con matriz:

[cos(th) -sin(th) 0]

[sin(th) cos(th) 0]

[0 0 1]

- Traslación a lo largo del eje x (por la longitud del eslabón)

Esto produce la matriz local:

A_p(:,:,i) = [cos(th_i_p) -sin(th_i_p) 0 l_i_p;

sin(th_i_p) cos(th_i_p) 0 0;

0 0 1 0;

0 0 0 1]

La traslación ocurre en x, lo que refleja un movimiento en el plano XY.

```
%
% Definición de las matrices locales para el robot planar:
P_p(:,:,1) = [l1_p; 0; 0];
R_p(:,:,1) = [cos(th1_p) -sin(th1_p) 0;
              sin(th1_p)  cos(th1_p) 0;
              0          0          1];

P_p(:,:,2) = [l2_p; 0; 0];
R_p(:,:,2) = [cos(th2_p) -sin(th2_p) 0;
              sin(th2_p)  cos(th2_p) 0;
              0          0          1];

P_p(:,:,3) = [l3_p; 0; 0];
R_p(:,:,3) = [cos(th3_p) -sin(th3_p) 0;
              sin(th3_p)  cos(th3_p) 0;
              0          0          1];
```

Transformación Global (Planar)

Se encadenan las transformaciones locales para obtener T_p(:,:,i)

La matriz global T_p(:,:,i) representa la posición y orientación del eslabón i respecto al marco inercial.

En el robot planar, T_p resultará en una posición en el plano XY y una orientación que es la suma de las rotaciones.

```
Vector_Zeros = zeros(1, 3);
A_p(:,:,GDL) = simplify([R_p(:,:,GDL) P_p(:,:,GDL); Vector_Zeros 1]);
T_p(:,:,GDL) = simplify([R_p(:,:,GDL) P_p(:,:,GDL); Vector_Zeros 1]);
PO_p(:,:,GDL) = P_p(:,:,GDL); % Posición local
RO_p(:,:,GDL) = R_p(:,:,GDL); % Orientación local

for i = 1:GDL
    A_p(:,:,i) = simplify([R_p(:,:,i) P_p(:,:,i); Vector_Zeros 1]);
    try
       T_p(:,:,i) = T_p(:,:,i-1) * A_p(:,:,i);
```

```
    catch
       T_p(:,:,i) = A_p(:,:,i);
    end
    disp(['Matriz de Transformación global T_p', num2str(i), ':'])
    T_p(:,:,i) = simplify(T_p(:,:,i));
    pretty(T_p(:,:,i)) % Visualización de la matriz global planar

    RO_p(:,:,i) = T_p(1:3,1:3,i);
    PO_p(:,:,i) = T_p(1:3,4,i);
end
```

```
Matriz de Transformación global T_p1:
/ cos(th1_p(t)), -sin(th1_p(t)), 0, l1_p \
|                                        |
| sin(th1_p(t)),  cos(th1_p(t)), 0,   0  |
|                                        |
|       0,              0,       1,   0  |
|                                        |
\       0,              0,       0,   1  /
Matriz de Transformación global T_p2:
/ cos(th1_p(t) + th2_p(t)), -sin(th1_p(t) + th2_p(t)), 0, l1_p + l2_p cos(th1_p(t)) \
|                                                                                    |
| sin(th1_p(t) + th2_p(t)),  cos(th1_p(t) + th2_p(t)), 0,     l2_p sin(th1_p(t))     |
|                                                                                    |
|             0,                         0,             1,             0            |
|                                                                                    |
\             0,                         0,             0,             1            /
Matriz de Transformación global T_p3:
/ #2, -#1, 0, l1_p + l2_p cos(th1_p(t)) + l3_p cos(th1_p(t) + th2_p(t)) \
|                                                                        |
| #1,  #2, 0,     l2_p sin(th1_p(t)) + l3_p sin(th1_p(t) + th2_p(t))     |
|                                                                        |
|  0,  0,  1,                              0                             |
|                                                                        |
\  0,  0,  0,                              1                             /

where

   #1 == sin(th1_p(t) + th2_p(t) + th3_p(t))

   #2 == cos(th1_p(t) + th2_p(t) + th3_p(t))
```

Comparacion (Planar):

Las matrices locales del robot planar son simples y reflejan una traslación en x y rotación en z resultando en una cinemática directa muy sencilla en el plano XY.

========== ROBOT ANTROPOMÓRFICO DE 3 GDL ==========

Transformación Local (Antropomórfico)

En el robot antropomórfico, la configuración es distinta:

- En la Articulación 1 se tiene:

P_a(:,:,1) = [0; 0; l1_a] --> Traslación a lo largo del eje z

R_a(:,:,1) = [cos(th1_a)   0   sin(th1_a);

sin(th1_a) 0 -cos(th1_a);

0 1 0]

- En las Articulaciones 2 y 3 se realiza la traslación en el plano XY, similar a la planar

Comparaciós: La diferencia principal es la primera transformación, que mueve el sistema en z y rota en función de un eje distinto.

```
% Definición de las matrices locales para el robot antropomórfico:
P_a(:,:,1) = [0; 0; l1_a];
R_a(:,:,1) = [cos(th1_a)  0  sin(th1_a);
              sin(th1_a)  0 -cos(th1_a);
              0           1      0];

P_a(:,:,2) = [l2_a*cos(th2_a); l2_a*sin(th2_a); 0];
R_a(:,:,2) = [cos(th2_a) -sin(th2_a) 0;
              sin(th2_a)  cos(th2_a) 0;
              0           0          1];

P_a(:,:,3) = [l3_a*cos(th3_a); l3_a*sin(th3_a); 0];
R_a(:,:,3) = [cos(th3_a) -sin(th3_a) 0;
              sin(th3_a)  cos(th3_a) 0;
              0           0          1];
```

Transformación Global (Antropomórfico)

Similar al planar, se encadenan las matrices locales para obtener T_a(:,:,i).

Debido a la primera transformación, la posición y orientación global serán diferentes.

```
%
A_a(:,:,GDL) = simplify([R_a(:,:,GDL) P_a(:,:,GDL); Vector_Zeros 1]);
T_a(:,:,GDL) = simplify([R_a(:,:,GDL) P_a(:,:,GDL); Vector_Zeros 1]);
PO_a(:,:,GDL) = P_a(:,:,GDL);
RO_a(:,:,GDL) = R_a(:,:,GDL);

for i = 1:GDL
    A_a(:,:,i) = simplify([R_a(:,:,i) P_a(:,:,i); Vector_Zeros 1]);
    try
       T_a(:,:,i) = T_a(:,:,i-1) * A_a(:,:,i);
    catch
       T_a(:,:,i) = A_a(:,:,i);
    end
    disp(['Matriz de Transformación global T_a', num2str(i), ':'])
    T_a(:,:,i) = simplify(T_a(:,:,i));
    pretty(T_a(:,:,i)) % Visualización de la matriz global antropomórfica

    RO_a(:,:,i) = T_a(1:3,1:3,i);
    PO_a(:,:,i) = T_a(1:3,4,i);
```

```
end
```

```
Matriz de Transformación global T_a1:
/ cos(th1_a(t)), 0,  sin(th1_a(t)),   0  \
|                                        |
| sin(th1_a(t)), 0, -cos(th1_a(t)),   0  |
|                                        |
|       0,       1,        0,       l1_a |
|                                        |
\       0,       0,        0,         1  /
Matriz de Transformación global T_a2:
/ cos(th1_a(t)) cos(th2_a(t)), -cos(th1_a(t)) sin(th2_a(t)),  sin(th1_a(t)), l2_a cos(th1_a(t)) cos(th2_a(
|
| cos(th2_a(t)) sin(th1_a(t)), -sin(th1_a(t)) sin(th2_a(t)), -cos(th1_a(t)), l2_a cos(th2_a(t)) sin(th1_a(
|
|        sin(th2_a(t)),                cos(th2_a(t)),              0,           l1_a + l2_a sin(th2_a(t))
|
\              0,                            0,                    0,                       1
Matriz de Transformación global T_a3:
/ cos(th1_a(t)) cos(#2), -cos(th1_a(t)) sin(#2),  sin(th1_a(t)),            cos(th1_a(t)) #1
|
| sin(th1_a(t)) cos(#2), -sin(th1_a(t)) sin(#2), -cos(th1_a(t)),            sin(th1_a(t)) #1
|
|        sin(#2),                cos(#2),               0,       l1_a + l2_a sin(th2_a(t)) + l3_a sin(#2)
|
\           0,                     0,                    0,                          1

where

   #1 == l2_a cos(th2_a(t)) + l3_a cos(#2)

   #2 == th2_a(t) + th3_a(t)
```

Comparacion (Antropomórfico):

La transformación local en la Articulación 1 involucra una traslación en z y una rotación que depende de th1_a,generando un marco que difiere del modelo planar. Las articulaciones 2 y 3 son similares al caso planar, pero la configuración global se ve afectada por la primera transformación.

========== CÁLCULO DEL JACOBIANO Y VELOCIDADES ==========

El cálculo del Jacobiano se realiza de forma similar para ambos robots,% pero el resultado refleja las diferencias en sus transformaciones locales y globales.

```
%% Robot Planar
% Jacobiano lineal (diferencial) para el robot planar:
Jv11_p = functionalDerivative(PO_p(1,1,GDL), th1_p);
Jv12_p = functionalDerivative(PO_p(1,1,GDL), th2_p);
Jv13_p = functionalDerivative(PO_p(1,1,GDL), th3_p);
Jv21_p = functionalDerivative(PO_p(2,1,GDL), th1_p);
Jv22_p = functionalDerivative(PO_p(2,1,GDL), th2_p);
Jv23_p = functionalDerivative(PO_p(2,1,GDL), th3_p);
Jv31_p = functionalDerivative(PO_p(3,1,GDL), th1_p);
Jv32_p = functionalDerivative(PO_p(3,1,GDL), th2_p);
Jv33_p = functionalDerivative(PO_p(3,1,GDL), th3_p);
```

```
jv_d_p = simplify([Jv11_p Jv12_p Jv13_p;
                   Jv21_p Jv22_p Jv23_p;
                   Jv31_p Jv32_p Jv33_p]);
disp('Jacobiano lineal obtenido de forma diferencial (Planar):')
```

```
Jacobiano lineal obtenido de forma diferencial (Planar):
```

```
pretty(jv_d_p)
```

```
/ - l2_p sin(th1_p(t)) - l3_p sin(th1_p(t) + th2_p(t)), -l3_p sin(th1_p(t) + th2_p(t)), 0 \
|                                                                                        |
|  l2_p cos(th1_p(t)) + l3_p cos(th1_p(t) + th2_p(t)),   l3_p cos(th1_p(t) + th2_p(t)), 0 |
|                                                                                        |
\                         0,                                          0,               0 /
```

```
% Jacobiano analítico para el robot planar:
Jv_a_p(:,GDL) = PO_p(:,:,GDL);
Jw_a_p(:,GDL) = PO_p(:,:,GDL);
for k = 1:GDL
    if RP_planar(k)==0
       try
          Jv_a_p(:,k) = cross(RO_p(:,3,k-1), PO_p(:,:,GDL) - PO_p(:,:,k-1));
          Jw_a_p(:,k) = RO_p(:,3,k-1);
       catch
          Jv_a_p(:,k) = cross([0,0,1], PO_p(:,:,GDL));
          Jw_a_p(:,k) = [0,0,1];
       end
    else
       % Para juntas prismáticas (no se aplican aquí)
       try
          Jv_a_p(:,k) = RO_p(:,3,k-1);
       catch
          Jv_a_p(:,k) = [0,0,1];
       end
       Jw_a_p(:,k) = [0,0,0];
    end
end
Jv_a_p = simplify(Jv_a_p);
Jw_a_p = simplify(Jw_a_p);
disp('Jacobiano analítico obtenido (Planar) - Lineal:')
```

```
Jacobiano analítico obtenido (Planar) - Lineal:
```

```
pretty(Jv_a_p)
```

```
/              #2,                          #2,           -l3_p sin(th1_p(t) + th2_p(t)) \
|                                                                                        |
| l1_p + l2_p cos(th1_p(t)) + #1, l2_p cos(th1_p(t)) + #1,               #1              |
|                                                                                        |
\               0,                           0,                          0             /

where
```

```
   #1 == l3_p cos(th1_p(t) + th2_p(t))

   #2 == - l2_p sin(th1_p(t)) - l3_p sin(th1_p(t) + th2_p(t))
```

```
disp('Jacobiano analítico obtenido (Planar) - Angular:')
```

```
Jacobiano analítico obtenido (Planar) - Angular:
```

```
pretty(Jw_a_p)
```

```
/ 0, 0, 0 \
|         |
| 0, 0, 0 |
|         |
\ 1, 1, 1 /
```

```
% Velocidades del efector final para el robot planar:
V_p = simplify(Jv_a_p * Qp_planar.');
disp('Velocidad lineal obtenida (Planar):')
```

```
Velocidad lineal obtenida (Planar):
```

```
pretty(V_p)
```

```
/                                      d                 d                       d
|                                - #2 -- th1_p(t) - #2 -- th2_p(t) - l3_p #3 -- th3_p(t)
|                                     dt                dt                      dt
|
|                                       d                                                 d
| (l1_p + l2_p cos(th1_p(t)) + l3_p #1) -- th1_p(t) + (l2_p cos(th1_p(t)) + l3_p #1) -- th2_p(t) + l3_p #1
|                                       dt                                                dt
|
\                                                        0
```

```
where

   #1 == cos(th1_p(t) + th2_p(t))

   #2 == l2_p sin(th1_p(t)) + l3_p #3

   #3 == sin(th1_p(t) + th2_p(t))
```

```
W_p = simplify(Jw_a_p * Qp_planar.');
disp('Velocidad angular obtenida (Planar):')
```

```
Velocidad angular obtenida (Planar):
```

```
pretty(W_p)
```

```
/                   0                   \
|                                       |
|                   0                   |
|                                       |
|  d             d             d        |
| -- th1_p(t) + -- th2_p(t) + -- th3_p(t) |
\ dt            dt            dt       /
```

```
%% Robot Antropomórfico
```

```
% Jacobiano lineal (diferencial) para el robot antropomórfico:
Jv11_a = functionalDerivative(PO_a(1,1,GDL), th1_a);
Jv12_a = functionalDerivative(PO_a(1,1,GDL), th2_a);
Jv13_a = functionalDerivative(PO_a(1,1,GDL), th3_a);
Jv21_a = functionalDerivative(PO_a(2,1,GDL), th1_a);
Jv22_a = functionalDerivative(PO_a(2,1,GDL), th2_a);
Jv23_a = functionalDerivative(PO_a(2,1,GDL), th3_a);
Jv31_a = functionalDerivative(PO_a(3,1,GDL), th1_a);
Jv32_a = functionalDerivative(PO_a(3,1,GDL), th2_a);
Jv33_a = functionalDerivative(PO_a(3,1,GDL), th3_a);

jv_d_a = simplify([Jv11_a Jv12_a Jv13_a;
                   Jv21_a Jv22_a Jv23_a;
                   Jv31_a Jv32_a Jv33_a]);
disp('Jacobiano lineal obtenido de forma diferencial (Antropomórfico):')
```

```
Jacobiano lineal obtenido de forma diferencial (Antropomórfico):
```

```
pretty(jv_d_a)
```

```
/ -sin(th1_a(t)) #1, -cos(th1_a(t)) #2,     -l3_a cos(th1_a(t)) #3    \
|                                                                     |
|  cos(th1_a(t)) #1, -sin(th1_a(t)) #2,     -l3_a sin(th1_a(t)) #3    |
|                                                                     |
\         0,                 #1,        l3_a cos(th2_a(t) + th3_a(t)) /

where

   #1 == l2_a cos(th2_a(t)) + l3_a cos(th2_a(t) + th3_a(t))

   #2 == l2_a sin(th2_a(t)) + l3_a #3

   #3 == sin(th2_a(t) + th3_a(t))
```

```
% Jacobiano analítico para el robot antropomórfico:
Jv_a_a(:,GDL) = PO_a(:,:,GDL);
Jw_a_a(:,GDL) = PO_a(:,:,GDL);
for k = 1:GDL
    if RP_antropomorfico(k)==0
       try
          Jv_a_a(:,k) = cross(RO_a(:,3,k-1), PO_a(:,:,GDL) - PO_a(:,:,k-1));
          Jw_a_a(:,k) = RO_a(:,3,k-1);
       catch
          Jv_a_a(:,k) = cross([0,0,1], PO_a(:,:,GDL));
          Jw_a_a(:,k) = [0,0,1];
       end
    else
       try
          Jv_a_a(:,k) = RO_a(:,3,k-1);
       catch
          Jv_a_a(:,k) = [0,0,1];
       end
       Jw_a_a(:,k) = [0,0,0];
```

```
    end
end
Jv_a_a = simplify(Jv_a_a);
Jw_a_a = simplify(Jw_a_a);
disp('Jacobiano analítico obtenido (Antropomórfico) - Lineal:')
```

```
Jacobiano analítico obtenido (Antropomórfico) - Lineal:
```

```
pretty(Jv_a_a)
```

```
/ -sin(th1_a(t)) #1, -cos(th1_a(t)) #2,     -l3_a cos(th1_a(t)) #3    \
|                                                                     |
|  cos(th1_a(t)) #1, -sin(th1_a(t)) #2,     -l3_a sin(th1_a(t)) #3    |
|                                                                     |
\         0,                 #1,        l3_a cos(th2_a(t) + th3_a(t)) /

where

   #1 == l2_a cos(th2_a(t)) + l3_a cos(th2_a(t) + th3_a(t))

   #2 == l2_a sin(th2_a(t)) + l3_a #3

   #3 == sin(th2_a(t) + th3_a(t))
```

```
disp('Jacobiano analítico obtenido (Antropomórfico) - Angular:')
```

```
Jacobiano analítico obtenido (Antropomórfico) - Angular:
```

```
pretty(Jw_a_a)
```

```
/ 0,  sin(th1_a(t)),  sin(th1_a(t)) \
|                                   |
| 0, -cos(th1_a(t)), -cos(th1_a(t)) |
|                                   |
\ 1,        0,              0       /
```

```
% Velocidades del efector final para el robot antropomórfico:
V_a = simplify(Jv_a_a * Qp_antropomorfico.');
disp('Velocidad lineal obtenida (Antropomórfico):')
```

```
Velocidad lineal obtenida (Antropomórfico):
```

```
pretty(V_a)
```

```
/                    d                                d                                                  d          \
| - sin(th1_a(t)) #1 -- th1_a(t) - cos(th1_a(t)) #2 -- th2_a(t) - l3_a cos(th1_a(t)) sin(#3) -- th3_a(t) |
|                    dt                               dt                                                 dt         |
|                                                                                                                    |
|                    d                                d                                                  d          |
|  cos(th1_a(t)) #1 -- th1_a(t) - sin(th1_a(t)) #2 -- th2_a(t) - l3_a sin(th1_a(t)) sin(#3) -- th3_a(t)  |
|                   dt                               dt                                                 dt         |
|                                                                                                                    |
|                                  d                               d                                                 |
|                             #1 -- th2_a(t) + l3_a cos(#3) -- th3_a(t)                                            |
\                                 dt                              dt                                                /

where

   #1 == l2_a cos(th2_a(t)) + l3_a cos(#3)
```

```
#2 == l2_a sin(th2_a(t)) + l3_a sin(#3)

#3 == th2_a(t) + th3_a(t)
```

```
W_a = simplify(Jw_a_a * Qp_antropomorfico.');
disp('Velocidad angular obtenida (Antropomórfico):')
```

```
Velocidad angular obtenida (Antropomórfico):
```

```
pretty(W_a)
```

```
/                /  d              d           \ \
|  sin(th1_a(t)) | -- th2_a(t) + -- th3_a(t) | |
|                \ dt             dt          / |
|                                               |
|                /  d              d           \ |
| -cos(th1_a(t)) | -- th2_a(t) + -- th3_a(t) | |
|                \ dt             dt          / |
|                                               |
|                   d                           |
|                  -- th1_a(t)                  |
\                  dt                          /
```

Análisis Comparativo General entre el Robot Planar y el Robot Antropomórfico

1. **Transformación Local:**

- **Robot Planar:**Cada transformación local se define mediante una rotación pura sobre el eje z y una traslación a lo largo del eje x. Esto se refleja en la matriz:

A_p(:;:;i) = [cos(theta_i) -sin(theta_i) 0 l_i

sin(theta_i) cos(theta_i) 0 0

0 0 1 0

0 0 0 1]

Este modelo es muy directo y simple, porque el desplazamiento ocurre únicamente en x y la rotación se realiza sobre z.

- **Robot Antropomórfico:**La primera articulación se modela con una traslación a lo largo del eje z y una rotación que involucra una componente en el eje y , mientras que las articulaciones 2 y 3 tienen traslaciones en el plano XY. Esto significa que la matriz local de la primera articulación es:

A_a(:;:;i) = [cos(theta_i) 0 sin(theta_i) 0

sin(theta_i) 0 -cos(theta_i) 0

0 1 0 l_i

0 0 0 1]

lo que introduce una complejidad adicional en el encadenamiento global de las matrices de transformaciones.

2. Transformación Global:

- **Robot Planar:**La transformación global se obtiene multiplicando secuencialmente las matrices locales. El resultado final define la posición y orientación del efector final en el plano XY de manera directa y sencilla, ya que la traslación se acumula en x y la orientación es la suma de las rotaciones en z.

- **Robot Antropomórfico:**Debido a la complejidad en la primera transformación (traslación en z y rotación con componente en y), el encadenamiento global produce una posición y orientación que difieren notablemente de las del robot planar. Esto afecta la forma en que se posiciona el efector final en el espacio y, por ende, su zona de trabajo.

Comparativa de matrices de transformacian globlas en planar y antropomorfico, para las 3 articulaciones:

	Articulacion 1	**Articulacion 2**
Planar	Matriz de Transformación global T_p1: / cos(th1_p(t)), -sin(th1_p(t)), 0, l1_p \ | | | sin(th1_p(t)), cos(th1_p(t)), 0, 0 | | | | 0, 0, 1, 0 | | | \ 0, 0, 0, 1 /	Matriz de Transformación global T_p2: / cos(th1_p(t) + th2_p(t)), -sin(th1_p(t) + | | sin(th1_p(t) + th2_p(t)), cos(th1_p(t) + | | 0, 0, | \ 0, 0,
Antropomorfico	Matriz de Transformación global T_a1: / cos(th1_a(t)), 0, sin(th1_a(t)), 0 \ | | | sin(th1_a(t)), 0, -cos(th1_a(t)), 0 | | | | 0, 1, 0, l1_a | | | \ 0, 0, 0, 1 /	Matriz de Transformación global T_a2: / cos(th1_a(t)) cos(th2_a(t)), -cos(th | | cos(th2_a(t)) sin(th1_a(t)), -sin(th | | sin(th2_a(t)), | \ 0,

3.Jacobiano y Velocidades:

- **Robot Planar:**La simplicidad de las transformaciones locales se refleja en un Jacobiano con estructura sencilla, donde el eje de rotación es constante [0 0 1]. Esto implica que el cálculo de las velocidades (lineal y angular) es directo, y las velocidades finales se obtienen mediante una multiplicación matricial que integra la suma de las rotaciones en z.
- **Robot Antropomórfico:**El Jacobiano, tanto en su forma diferencial como analítica, incorpora la complejidad de la primera transformación. Como resultado, el Jacobiano y las velocidades obtenidas muestran una mayor complejidad, lo que se traduce en una respuesta cinemática más adaptable, pero también más difícil de interpretar de manera intuitiva.

Comparativa de matrices Jacobianas en planar y antropomorfico:

	Jacabion lineal (Diferencial)
Planar	Jacobiano lineal obtenido de forma diferencial (Planar): / - l2_p sin(th1_p(t)) - l3_p sin(th1_p(t) + th2_p(t)), -l3_p sin(th1_p(t) + th2_p(t)), | | l2_p cos(th1_p(t)) + l3_p cos(th1_p(t) + th2_p(t)), l3_p cos(th1_p(t) + th2_p(t)), | \ 0, 0,
Antropomorfico	/ -sin(th1_a(t)) #1, -cos(th1_a(t)) #2, -l3_a cos(th1_a(t)) #3 \ | | cos(th1_a(t)) #1, -sin(th1_a(t)) #2, -l3_a sin(th1_a(t)) #3 | | \ 0, #1, l3_a cos(th2_a(t) + th3_a(t)) / where #1 == l2_a cos(th2_a(t)) + l3_a cos(th2_a(t) + th3_a(t)) #2 == l2_a sin(th2_a(t)) + l3_a #3 #3 == sin(th2_a(t) + th3_a(t))

Comparativa de velocidades lineales y angulares en planar y antropomorficos:

	Velocidad lineal

Planar

```
/                                    d                d                    d
|                              - #2 -- th1_p(t) - #2 -- th2_p(t) - l3_p #3 -- th3_p(t
|                                   dt               dt                   dt
|
|                                     d                                              d
| (l1_p + l2_p cos(th1_p(t)) + l3_p #1) -- th1_p(t) + (l2_p cos(th1_p(t)) + l3_p #1) --
|                                     dt                                             dt
|
\                                                    0

where

   #1 == cos(th1_p(t) + th2_p(t))

   #2 == l2_p sin(th1_p(t)) + l3_p #3

   #3 == sin(th1_p(t) + th2_p(t))
```

Antropomorfico

```
/                     d                              d
| - sin(th1_a(t)) #1 -- th1_a(t) - cos(th1_a(t)) #2 -- th2_a(t) - l3_a cos(th1_a(t)) si
|                    dt                             dt
|
|                     d                              d
|  cos(th1_a(t)) #1 -- th1_a(t) - sin(th1_a(t)) #2 -- th2_a(t) - l3_a sin(th1_a(t)) sin
|                    dt                             dt
|
|                                  d                           d
|                             #1 -- th2_a(t) + l3_a cos(#3) -- th3_a(t)
\                                 dt                          dt

where

   #1 == l2_a cos(th2_a(t)) + l3_a cos(#3)

   #2 == l2_a sin(th2_a(t)) + l3_a sin(#3)

   #3 == th2_a(t) + th3_a(t)
```

4.Conclusión General:

- El **robot planar** presenta una cinemática sencilla y directa, ideal para aplicaciones en las que el movimiento se limita al plano XY. La definición de sus transformaciones y el cálculo de su Jacobiano permiten una implementación y análisis más rápidos y fáciles.
- El **robot antropomórfico**, por otro lado, al incluir una transformación inicial con traslación en z y una rotación más compleja, ofrece una mayor destreza y capacidad de maniobra en el espacio. Sin embargo, esto conlleva un encadenamiento global y un cálculo del Jacobiano que reflejan dicha complejidad.